# Abstract (Zheng)

We proposed a network combined yolov3 and mask rcnn to do the instance segmentation task. We use darknet as feature map backbone, and yolov3’s original approach to get classes and bbox. Then align bbox on the feature maps and go through the mask fcn branch to get mask prediction.

Hopefully our network can be faster than mask rcnn and get high AP point. We train our network on COCO dataset.

# Introduction (Wang, Liu)

The state-of-art method Mask RCNN is really powerful and has elegant precisions for instance segmentation. However, the slow feature map generation process become a bottleneck when we try to increase the speed. Also, it’s based on two stages: region proposal network and detection network(classification, detection and mask prediction).

YOLOv3, as one of the trending framework in object detection, enjoys real-time speed with its unified structure and even higher precision with the pyramid of multi-scale feature maps. With similar performance on COCO dataset, YOLOv3-416 runs two times faster than RetinaNet-50-500[1]. It’s also observed that with spatial pyramid pooling method, the framework would be more robust to the scale change and occlusion[2]. By adding a mask branch, we incorporate this extreme fast object detection method in our instance segmentation network.

By combining these two framework, we introduce a new method to perform instance segmentation with high precision.

# Related Works (Wang, Liu)

Instance Segmentation

A lot of research effort has been made to push instance segmentation accuracy. Some methods first perform semantic segmentation followed by boundary detection [20], pixel clustering [3, 23], or learn an embedding to form instance masks [30, 15, 8, 11]. Mask-RCNN [16] as a representative two-stage instance segmentation approach, first generates candidate region-of-interests (ROIs), then generate mask for ROIs detected. Follow-up works try to improve its accuracy by enriching the FPN features [27] or addressing the incompatibility between a mask’s confidence score and its localization accuracy [18], which requires subsequent computations that make them unable to obtain real-time speeds (30 fps). In our work, we keep YOLO structure as our bounding box detection part and use only mask branch from Mask-RCNN for mask generation, to keep us away from large computation and save processing time as much as possible.

Real-time Instance Segmentation

While real-time object detection [28, 32, 33, 34], and semantic segmentation [2, 31, 10, 45] methods exist, few works have focused on real-time instance segmentation. Mask R-CNN [16] remains one of the fastest instance segmentation methods on semantically challenging datasets.

Among some works in real-time instance segmentation, straight to Shapes [19] performs instance segmentation with learned encodings of shapes at 30 fps, but its accuracy is far from that of modern baselines. With a light-weight backbone detector, Box2Pix [40] implements a hand-engineered algorithm to obtain 10.9 fps on Cityscapes [5] and 35 fps on KITTI. However, they report no results on the more challenging and semantically rich COCO dataset, which has 80 classes compared to the 8 of KITTI and Cityscapes. Also, there’s a large drop in performance going from a semantically simple dataset (KITTI) to a more complex one (Cityscapes), so the performance on an even more difficult dataset (COCO) is to be questioned.

Mask RCNN

The proposal-based method is most popular in instance segmentation, which have a strong connection to object detection. Mask R-CNN extends Faster R-CNN by adding a branch for predicting segmentation masks on each Region of Interest (RoI), in parallel with the existing branch for classiﬁcation and bounding box regression.

They predict an m × m mask from each RoI using FCN, which allows each layer in the mask branch to maintain the explicit m × m object spatial layout without collapsing it into a vector representation that lacks spatial dimensions. Also, the FCN requires less parameters compared with fc layers.

Before passing the RoI into the FCN, instead of RoIPool, the Mask RCNN applies RoIAlign to properly align the extracted features with the input. In this way, they avoid any quantization of the RoI boundaries or bins.

YOLO

YOLO is an end-to-end object detection framework that runs in real time. The first version comes out in 2016, which is quite impressive with its real-time performance. Then we have YOLOv2 and YOLOv3, which takes YOLO to a new level with its even higher precision, especially when detecting small objects.

Aside from a structure called Darknet-53, which originally has 53 layer network trained on ImageNet, YOLOv3 also incorporates residual skip connections, upsampling, and makes detections at three different scales, which helps address the issue of detecting small objects. Our work is based on YOLOv3, and explores several morphing structures like YOLOv3-tiny and YOLOv3-spp. The latter one, YOLOv3-spp is especially robust to objects in different scales with its multiple effective field-of-views. Based on the YOLOv3 part, we add a mask branch to turn the object detection problem into an instance segmentation task.

# YOLO-MASK model (Zheng)

Inspired from mask rcnn to build a multi-task learning using multiple fully connected layer.Our goal is to add a mask branch to an existing one-stage object detection model in the same vein as Mask R-CNN does to Faster R-CNN. We want to add another branch on yolo to do the instance segmentation task.

A good feature map backbone is base for tasks like classifier, bbox perdition and instance segmentation. In mask rcnn, the author proposed resnext and pyramid feature network to give fully convolutional network better feature maps.the better backbone alone improves AP by 4 point. And YOLOv3 purposed darknet59 which has deeper networks. It has multi scale layer structure like feature pyramid network. Compare with mask rcnn, YOLOv3 can provide more accurate bbox proposal and higher speed. Hopefully more accuracy bbox can give us higher AP point on instance segmentation task.

**Feature map**

We use darknet59 which proposed in yolov3 as our backbone network to extract feature. The darknet59 framework looks like fig 1.

Darknet59 is a deeper network than previous version of yolo. It contains 106 fully convolutional layers. This deeper network does slower the algorithm, but it gives us better feature maps. It also has multi scale layer structure like feature pyramid network.Darknet59 introduced the idea of ResidualNet, which inclues 3x3 convolution kernel, 1x1 convolution kernel. The newer architecture boasts of residual skip connections, and up sampling. The most salient feature of darknet59 is that it makes detections at three different scales which is similar to the mask rcnn’s pyramid feature network. By upsampling the lower layer and adding it with previous layer, the darknet59 has the down-to-top and top-to-down structure, which can have better performance on detecting different size object in multiple scales.

It can extract feature in the image faster than ResNet but with high quality feature maps. The multi-scale feature map is merged element-wide by adding. In this way, we can get more high-level semantic information from the later layers, and also get fine-grained information from the pervious layer.

**Bbox prediction and classification**

For the bounding box prediction and classification part, we follow the algorithm in yolov3. the output bbox and class label are generated by applying a 1 x 1 kernel on a feature map. The detection is done by applying 1 x 1 detection kernels on feature maps of three different sizes at three different scales in the network. The size of the smallest feature map is (13x13), which can detect big object in the image. The scale 2 is (26x26) size feature map, which used to detect medium size object. The last scale size is (52x52), which can detect small objects in the picture very well.

The shape of the detection kernel is 1 x 1 x (B x (5 + C) ). Here B is the number of bounding boxes a cell on the feature map can predict, “5” is for the 4 bounding box attributes and one object confidence, and C is the number of classes. In YOLO v3 trained on COCO, B = 3 and C = 80, so the kernel size is 1 x 1 x 255. We predict 3 boxes per scale. Applying convolution layers on each scales’ feature map shrinks it to the size of predition feature map which is NxNx[Bx(4+1+80)].

The first detection is made by the 82nd layer. For the first 81 layers, the image is down sampled by the network, such that the 81st layer has a stride of 32. If we have an image of 416 x 416, the resultant feature map would be of size 13 x 13. One detection is made here using the 1 x 1 detection kernel, giving us a detection feature map of 13 x 13 x 255.

Then, the feature map from layer 79 is subjected to a few convolutional layers before being up sampled by 2x to dimensions of 26 x 26. This feature map is then depth concatenated with the feature map from layer 61. Then the combined feature maps is again subjected a few 1 x 1 convolutional layers to fuse the features from the earlier layer (61). Then, the second detection is made by the 94th layer, yielding a detection feature map of 26 x 26 x 255.

A similar procedure is followed again, where the feature map from layer 91 is subjected to few convolutional layers before being depth concatenated with a feature map from layer 36. Like before, a few 1 x 1 convolutional layers follow to fuse the information from the previous layer (36). We make the final of the 3 at 106th layer, yielding feature map of size 52 x 52 x 255.

In bbox prediction, we use 9 anchor boxes. Three for each scale. Then, arrange the anchors is descending order of a dimension. Assign the three biggest anchors for the first scale , the next three for the second scale, and the last three for the third.

The number of bbox prediction is 3 x (13x13+26x26+52x52) = 10,648. Then we do the NMS to select the bbox. After selecting the bbox, we rescale the bbox to original image size as the bbox proposal for mask branch prediction.

**RoIAlign**

**FCN mask branch**

And we add a fully connected branch on the layer before upsampling which used as our mask prediction feature map. We can know which layer we should use based on YOLOv3 classes and bbox prediction. We align predicted bbox on that feature map layer. Then put that feature map into mask branch to get mask prediction. The structure of mask branch also followed the design of mask rcnn shows in fig 2. RoIAlign generates a feature map with size of 14x14. Then make a deconvolution to increase the feature map to 28x28, and take sigmoid for each classes. The output is a binary mask with size of 28x28. At last, transform the 28x28 mask using bilinear interpolation to the size of the box in the original image.

# Experiment

We first try to train our network end-to-end, but as mask branch is highly depended on the bbox proposed by backbone network, the training tends to be very unstable. So we choose to train our network in 2 steps. In the first stage, we only train our darknet to predict bbox and labels. After the 200+ epoch, we can get about 0.57 AP for bbox. Then we froze the weight of darknet. So our mask branch can get a good RoI proposal. When we train the mask branch, the batch size must set as 1. Because the CropAndResize function can only run on single batch.

When we run the experiment, we found that the mask size 28x28 maybe too large for the darknet. Because the smallest feature map only has 13x13. So we set stride in last layer as 4, so we can get a 7x7 mask as out prediction. The speed of our network reached our goal. One forward for darknet only takes about 0.002s per image and the mask branch takes only 0.008s per image.

# Conclusion

We achieve high speed with this one-stage instance segmentation framework. As for mask branch, it reaches 125 fps on COCO2014 validation dataset, while the detector part could generate bounding boxes at the speed of 0.002 per image. The processing time is reasonable for a real-time instance segmentation. However, we fall behind state-of-the-art instance segmentation methods in overall performance, especially the accuracy term. This mainly caused by errors in detection part, which is Darknet here.

7. Reference

[1]Redmon, J. and Farhadi, A., 2018. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

[2]Kim, K.J., Kim, P.K., Chung, Y.S. and Choi, D.H., 2018, November. Performance Enhancement of YOLOv3 by Adding Prediction Layers with Spatial Pyramid Pooling for Vehicle Detection. In *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)* (pp. 1-6). IEEE.

[20]A. Kirillov, E. Levinkov, B. Andres, B. Savchynskyy, and C. Rother. Instancecut: from edges to instances with multicut. In CVPR, 2017.

[3] M. Bai and R. Urtasun. Deep watershed transform for instance segmentation. In CVPR, 2017. 2

[23] X. Liang, L. Lin, Y. Wei, X. Shen, J. Yang, and S. Yan. Proposal-free network for instance-level object segmentation. TPAMI, 2018.

[30] A. Newell, Z. Huang, and J. Deng. Associative embedding: End-to-end learning for joint detection and grouping. In NeurIPS, 2017.

[15] A. W. Harley, K. G. Derpanis, and I. Kokkinos. Segmentation-aware convolutional networks using local attention masks. In ICCV, 2017.

[8] B. De Brabandere, D. Neven, and L. Van Gool. Semantic instance segmentation with a discriminative loss function. arXiv preprint arXiv:1708.02551, 2017.

[11] A. Fathi, Z. Wojna, V. Rathod, P. Wang, H. O. Song, S. Guadarrama, and K. P. Murphy. Semantic instance segmentation via deep metric learning. arXiv preprint arXiv:1703.10277, 2017.

[16] K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask r-cnn. ´ In ICCV, 2017.

[27] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia. Path aggregation network for instance segmentation. In CVPR, 2018.

[18] Z. Huang, L. Huang, Y. Gong, C. Huang, and X. Wang. Mask scoring r-cnn. In CVPR, 2019.

[28] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In ECCV, 2016.

[32] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In CVPR, 2016.

[33] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In CVPR, 2017.

[34] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. arXiv:1804.02767, 2018.

[2] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. CoRR, 2015.

[31] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. CoRR, 2016.

[10] N. Dvornik, K. Shmelkov, J. Mairal, and C. Schmid. Blitznet: A real-time deep network for scene understanding. In ICCV, 2017.

[45] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia. Icnet for real-time semantic segmentation on high-resolution images. In ECCV, 2018.

[16] K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask r-cnn. ´ In ICCV, 2017.

[19] S. Jetley, M. Sapienza, S. Golodetz, and P. H. Torr. Straight to shapes: real-time detection of encoded shapes. In CVPR, 2017.

[40] J. Uhrig, E. Rehder, B. Frohlich, U. Franke, and T. Brox. ¨ Box2pix: Single-shot instance segmentation by assigning pixels to object boxes. In IEEE Intelligent Vehicles Symposium, 2018.

[5] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In CVPR, 2016.

[7] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick. Microsoft coco: Common objects in context. In ECCV, 2014.

Ren S, He K, Girshick R, et al. Faster r-cnn: Towards real-time object detection with region proposal networks[C]//Advances in neural information processing systems. 2015: 91-99.

Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 3431-3440.